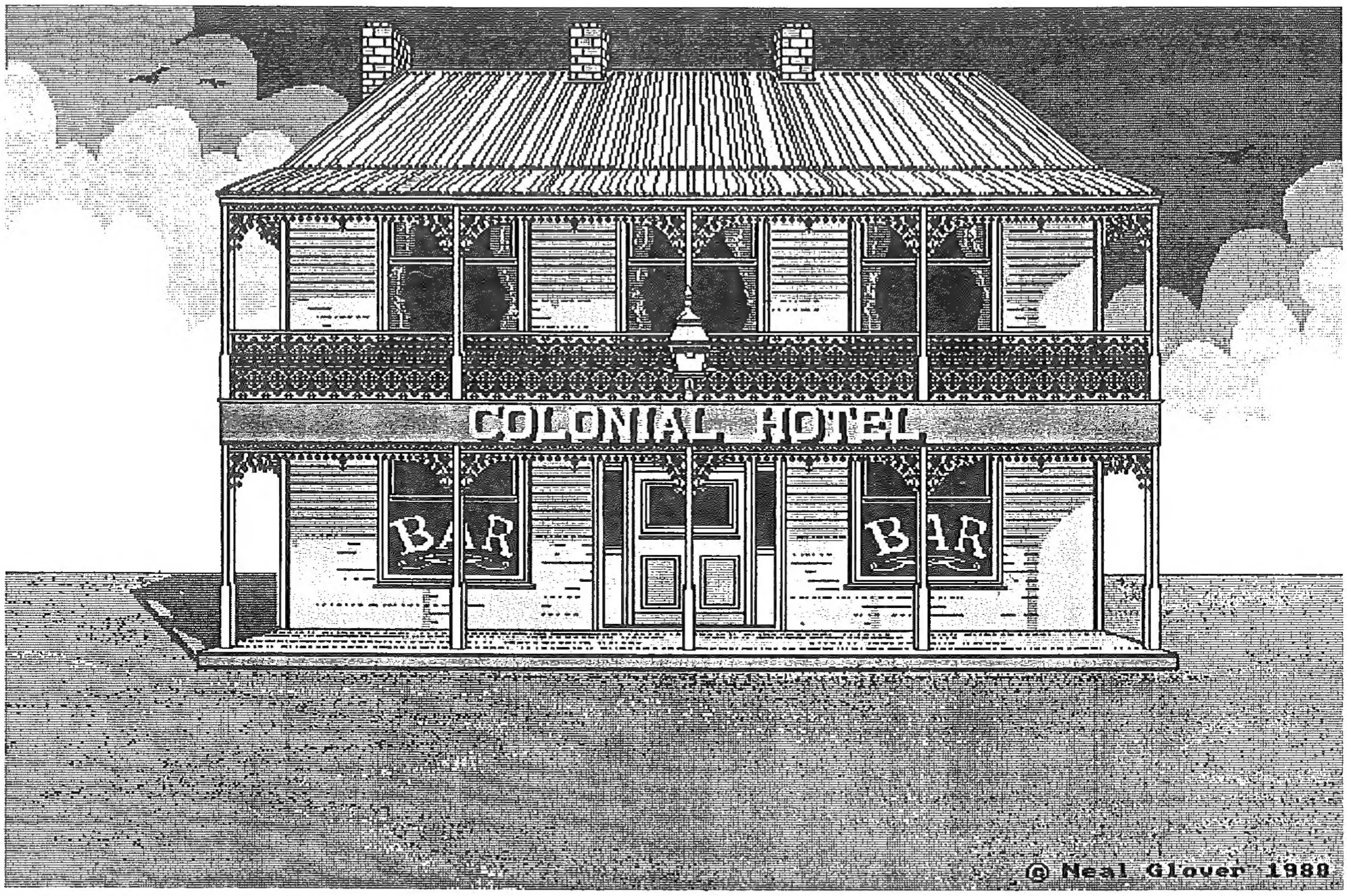
AMIGA

WORKBENCH

$2 RRP

Registered by Australia Post — Publication No. VBG7930

Number 27 Circulation: 1200 August 1988



Next AUG Meeting

Sunday, August 21st, 1988 at 2pm

(Doors open at 1pm, meeting starts at 2pm sharp)

AUG meetings are held in the Rotunda at Monash University
Wellington Road, Clayton Melways map 70 reference F10 and map 84A

Amiga Users Group Inc, PO Box 48, Boronia, 3155, Victoria, Australia

AMIGA™ Users Group

Who Are We?

The Amiga Users Group is a non-profit association of people interested in the Amiga computer and related topics. With over 1000 members, we are the largest independent association of Amiga users in Australia.

Club Meetings

Club meetings are held at 2pm on the third Sunday of each month in the Rotunda at Monash University, Wellington Road, Clayton. Details on how to get there are on the back cover of this newsletter. The dates of upcoming meetings are:

Sunday, August 21st at 2pm
Sunday, September 18th at 2pm
Sunday, October 16th at 2pm

Production Credits

This month's newsletter was edited by Peter Jetson. Equipment and software used was: Non-descript Taiwanese PC Clone computer, Brother HR-40 printer, Brother HL-8 printer, Gemini 10x printer, Wordstar, Fancy Font and Grabbit.

Copyright and Reprint Privileges

Contributions

Articles, papers, letters, drawings and cartoons are actively sought for publication in Amiga Workbench. Please submit your contributions on disk, since that means they don't have to be re-typed! All disks will be returned! Please save your article in **text-only** format (If it can be loaded by ED, it is text-only). Absolute deadline for articles is 16 days before the meeting date. Contributions can be sent to: The Editor, AUG, PO Box 48, Boronia, 3155.

Membership and Subscriptions

Membership of the Amiga Users Group is available for an annual fee of $20. To become a member of AUG, fill in the membership form in this issue (or a photocopy of it), and send it with a cheque for $20 to:

Amiga Users Group, PO Box 48, Boronia, 3155

Public Domain Software

Disks from our public domain library are available on quality 3.5" disks for $8 each including postage on AUG supplied disks, or $2 each on your own disks. The group currently holds over 180 volumes, mostly sourced from the USA, with more on the way each month. Details of latest releases are printed in this newsletter, and a catalog disk is available.

Member's Discounts

The **Amiga Users Group** negotiates discounts for its members on hardware, software and books.

Currently, **Technical Books** in Swanston Street in the city offers **AUG** members a 10% discount on computer related books, as does **McGills** in Elizabeth Street. Just show your membership card. Although we have no formal arrangements with other companies yet, most seem willing to offer a discount to AUG members. It always pays to ask!

Back Issues of Newsletter

All back issues of Amiga Workbench are now available, for $2 each including postage. Note that there may be delays while issues are reprinted. Back Issues are also available at meetings.

AmigaLink - Our Bulletin Board System

The Amiga Users Group operates a bulletin board system devoted to the Amiga, using the Opus message and conferencing system. AmigaLink is available 24 hours a day on (03) 792 3918, and can be accessed at V21 (300bps), V22 (1200bps), V23 (1200/75bps) or V22bis (2400bps) using 8 data bits, 1 stop bit and no parity.

AmigaLink is part of the world-wide Fido/Opus network of bulletin boards, and we participate in the national and international Amiga conferences. Amiga Link has selected Public Domain software available for downloading, and encourages the uploading of useful public domain programs from its users. Amiga Link is FidoNet node number 631/324.

Newsletter Advertising

The Amiga Users Group accepts commercial advertising in Amiga Workbench subject to the availability of space at these rates:

Quarter page	$20
Half page	$40
Full page	$70
Double page spread	$120

These rates are for full-size camera-ready copy **only**. We have no photographic or typesetting facilities. Absolute deadline for copy is 16 days before the meeting date. Send the copy and your cheque to: The Editor, AUG, PO Box 48, Boronia, 3155, Victoria.

Amiga Users Group Committee

Co-ordinator:	Bob Scarfe	376 4143	Kensington
Vice Co-ord:	Lester McClure	??? ????	Mt Waverley
Meeting Chair:	Ron Wail	??? ????	Doncaster
Secretary:	John Elston	375 4142	Moonee Ponds
Treasurer:	Donna Heenan	729 2327	Bayswater
Membership:	Neil Murray	794 5683	Dandenong
Purchasing:	Bohdan Ferens	792 1138	Dandenong
Book Library:	Joan Wood	580 7463	Aspendale
Disk Library:	Craig Hutchison	578 3962	Carnegie
Editor:	Peter Jetson	762 1386	Boronia
Committee:	Doug Myers	428 0682	Abbotsford

AMIGA WORKBENCH TeX REVIEW

Lachlan Myers

Typesetting on the Amiga - with help from Donald Knuth

(Alex got me into this; it's really his fault.)

Somehow, he caught TeX from a PC and has now infected my Amiga. Worse still, I think my brain has been subverted and TeX has taken over from the usual day to day concerns (like "Are my coins the right shape for parking meters?" or "is there life south of the Yarra?" – you know the kind of thing). AmigaTeX comes from Stanford USA, via Alex and TeXworks, who can provide local TeX for all kinds of computers, even I?M. Don't take my word for it; ask him (tel (03) 699-4083)!

As most of you know, Stanford is a university in America; American Universities are not all degree factories, with courses in garbology and the Social Implications of Food Processors – some of them actually do real intellectual work.

Centres of learning like CalTech, Stanford, and MIT have allowed great thinkers like Marvin Minsky to consider Life, the Universe, and Very Expensive Computers as Toys. These are mega-smart guys, with brains the size of small asteroids, paid to be clever and do clever things, without too many restrictions. (The results are often impressive, and if you don't believe that, read Prof. Minsky's latest, *"Society of Mind"*.)

Stanford University allows Donald Knuth this kind of luxury, which he has used to write three extremely clever books about *The Art of Computer Programming*, which details and discusses the basic algorithms used in computer programs. As well, he writes all kinds of mathematical articles (to entertain and enlighten mathematicians, typesetters, and other odd-balls).

But the laying out of the first three volumes of his masterwork, and getting everything just so was a very frustrating business, so D.E.K sat down and thought most hard about it all. And it seemed to him that the handling of text was just the sort of thing that a Stanford professor should be concerned with, and try to sort out. So he did, and the result, after eight years is TeX and its related programs. Well, he did a *little* more; designed a new way of writing and documenting computer programs, too.

Don't confuse TeX and that other yuppie status symbol – desktop publishing – there is no comparison. The first does *magnificent* things with text, like beautiful books; the other does useful things with text, like newsletters. In typesetting there is a craft, and an art. In desktop publishing there is a skill, and a result. To illustrate this, take the special typesetting rules for dealing with "." the humble full-stop, or period, and the double quotes left and right.

You can do some of this in programs like Pagemaker, using the fonts available from Adobe, (Pagemaker, and similarly, Adobe are trademarks† of their respective trademarkers, and who cares?) but most people won't know where to begin. It is necessary to have looked around the program quite a bit to find them in the first place, and be aware of the typesetting conventions for ligatures and the kerning for full-stops before setting out the document. In fact, that is the whole trouble with desktop publishing; if you aren't an expert, then a good wordprocessing program is more useful, and will be necessary anyway (just try using the editor in Pagemaker for more than corrections). What you then do is sort of pour the text into the columns like honey into jars, and woe betide you if you misjudge it! It can take hours to unstick!

TeX will handle these either automatically in the simple cases, or accept explicit direction from you, the user, at any time. So when the use is different in Mr. Smith, in the three dots (...) of an ellipsis (don't you see ... ?), or the description of a mathematical series $(x_1+x_2+x_3 \ldots x_n)$ and at the end of a sentence, then you can do it. Double quotes are different left – " and right – ", as can be seen in the preceding paragraph.

Do ligatures keep you awake at night? These cute little items are there to trip up the uninitiated – but TeX will catch them automatically! *¿No comprendo? ¡De nada!*

`ff`	ﬀ
`fi`	ﬁ
`ffl`	ﬄ

The academic heritage shows; I've seen a number of reviews for IBM-PC and Macintosh TeX packages which are clearly written by mathematicians, who are amazed that finally they are able to use their unconventional squiggles for everything. It isn't always set up for conventional typesetting – this can always be done, but may require digging a little deeper into the program.

So what does TeX do that's so fantastic?

1 It handles all the fiddly bits invisibly, from footnotes to chapter headings and bibliographic references and simple items for enumeration.

2 It produces properly set out typeset documents with minimum effort. Most DTP programs will handle standard typesetting conventions reasonably well, but won't deal with unusual problems. Mathematics has particular requirements, with lots of funny symbols and special spacing, subscripts on subscripts, and few DTP or WP packages will cope with these.

3 Any thing you want to do can be done, either easily through the inbuilt commands or by writing a macro. There is an active circulation of these through the TeX Users' Group (TUG), based in Providence, Rhode Island (the only acceptable

† Note: this article contains lots of names that are possibly trademarks (or not). You probably know who they are, they know who they are, but I've forgotten. Please fill in the appropriate company in your mind as you read this. The one you may not know is TeX – belonging to the American Mathematical Society (Gawdelpus).

Reds in the USA also hail from here). For example;

One common area that causes problems is tables, but not for TEX;

1985	1440	500
1986	1790	600
1987	2100	800

Just to test it out I decided to try a small trial, from *"Alice in Wonderland"*, which may be informative. The construction is not particularly sophisticated, as it is based on a tabular environment, but the result is quite satisfactory.

Fury said to
a mouse, That
he met
in the
house
'Let us
both go
to law:
I will
prosecute
you. —
Come, I'll
take no
denial;
we must
have a
trial:
for
really
this
morning
I've
nothing
to do.'
Said the
mouse to
the cur,
'Such a
trial
dear sir,
With no
jury or
judge,
would be
wasting
our breath.'
'I'll be
judge,
I'll be
jury,'
Said
cunning
old Fury;
I'll try
the whole
cause,
and
condemn
you
to
death'

CED - Amiga Text Editor

by Lester McClure

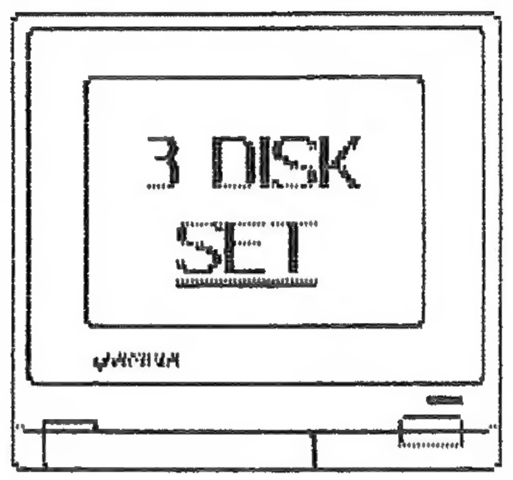
One of the most useful programs for any computer system is a general purpose text editor. Most systems are supplied with such a tool and the Amiga is no exception, with 'Ed' and 'Edit' hidden away in the C directory. Anyone who has used either of these programs will soon have found their limitations, although I still find Ed quite useful for a quick change to my startup-sequence.

If you decide, as I have, that the standard AmigaDos editors are not good enough, where do you start looking for alternatives? What do you want/expect an editor program to do? How much are you prepared to pay? If you decide to buy a commercial package, can you 'try before you buy'? This article outlines my experiences and how I found a solution to the above problems.

What should an editor do?

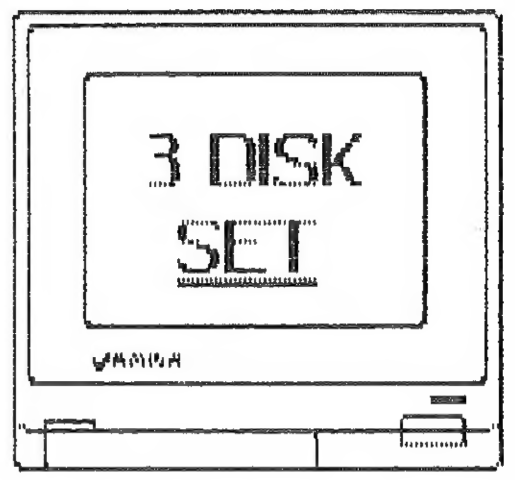
There are three things in this world people are guaranteed to 'forever disagree' about; religion, politics and text editors. There is NO best text editor, just one that suits your personal requirements better than others (a bit like politics and religion). I am prepared, however, to suggest that a general purpose text editor for the Amiga should do the following :-

- It must run from CLI or Workbench and fully support multitasking.
- It should be designed for use with the mouse for ease of learning, and the keyboard for speed.
- It must be intuitive to use, i.e. standard menus, gadgets and requesters should be used with the expected shortcut codes, for example A-q = Quit, A-s = search etc. The shift/control keys should modify standard key functions in a logical manner, e.g. Shift-Cursor Right should move to end of line, Shift-Cursor Down should move down a page/screen.
- There should be some form of on-line help facility to provide quick assistance without having to refer to the manual. More programs should take advantage of the existence of a HELP key on the keyboard.
- Function keys should be programmable and it must be possible to define sequences of commands as macros. These definitions also need to be saved or loaded to suit the file being edited.
- Multiple windows for editing and viewing different sections of the same file is a very useful feature especially if cutting and pasting between windows is allowed.
- It must support TEXT and PROGRAMMER modes with options like auto-indent for typing programs, and word-wrap for typing documents.
- I prefer an editor to operate in INSERT mode but others prefer OVERSTRIKE. There is no real advantage in one over the other so let's have both.
- Selectable tab-stops and also automatic spaces instead of tabs should be supported along with the ability to operate in an open page layout, editing mode. The cursor can then be moved anywhere on the screen and the text entered without any regard for 'end of line'. This is sometimes referred to as 'sparse space' editing.
- If the editor can provide additional processing features such as vertical columns and insertion of printer control codes, that's fine but for full

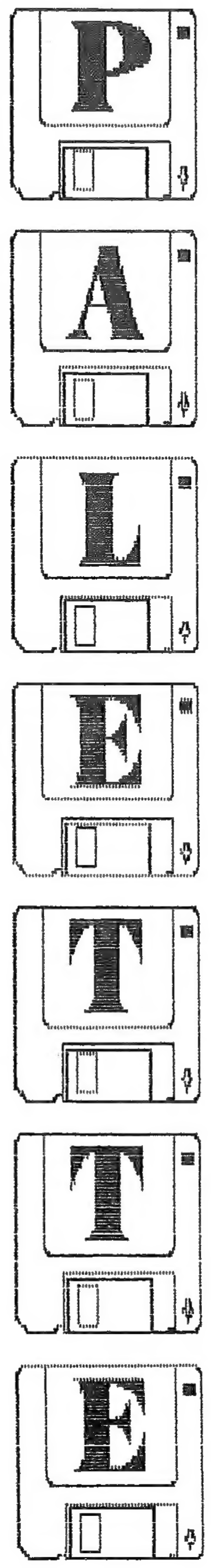


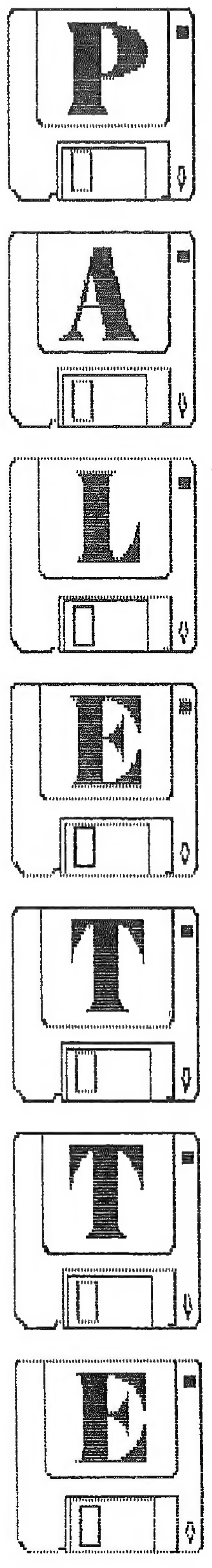
PALETTE®

GRAPHICS DISK-ZINE
FOR THE AMIGA®



Volume 1 Number 1
NOW available





An all Australian production that specializes in what the Amiga does best. . .

Graphics

THIS ISSUE:- Vol.1. No:1. AUGUST 1988.

PALETTE: All about the new graphics/DTV (DeskTop Video) disk-zine
E.A's PHOTOLAB: You can have a chance to own an original
PREFERENCES: How to set up your Workbench for PAL video usage
AMIGA AND PAL VIDEO: An in-depth article on improving your PAL graphics
DTV ARTICLES LISTING: A complete listing of Amiga graphic/DTV articles
EXPRESS PAINT V2.0: A review of PAR's new paint software
ANIMATION EFFECTS: A review of Hash Enterprise's DTV effects package
HINTS AND TIPS: Nine tricks of the trade also Digi-View Aerial Imaging
BRUSHES: A selection of hi-res brushes to use in your paint programs
PALETTE ANIMATION: A demo and how it was put together
BEST OF PD: Three graphic programs to add to your FX drawer

PLUS 25 LOW, MED, HIGH, HAM IMAGES TO IMPRESS
- - - BONUS DISK INCLUDED - - -
The first installment in PALETTE'S video serial - "MISSION : MICROBE"
- RECOMMENDED RETAIL PRICE: $26.00 -
(Add $2.00 Post + Packing Aust.)
Available From: PALETTE. 66 Parer St., Burwood. 3125. Victoria.
(Amiga is a registered trademark of Commodore-Amiga Inc.)

WYSIWYG operation, I expect to have to use a dedicated word processor package.

What Packages are Available?

If you remove from consideration all of the commercial desk top publishing and dedicated word processing packages, there is not a great deal left. I used Lattice Screen Editor (LSE) for quite some time, and while it was quite a good programming editor, it lacked any general word processing capabilities. It also had the feel of being a program that had been moved to the Amiga rather than written for the Amiga. Others that come to mind are TxEd and AEdit which, from reviews I've read, I believe would not offer me anything new. I am not prepared to spend more money just to be disappointed again once I've had a decent chance to try them out.

There are, of course, always the Public Domain offerings. The main contenders here are Dme, MicroEMACS and Uedit. I have tried each of these and found none of them individually had the right combination of features to suit me, although collectively they came pretty close. Then, tucked away on Fish disk #95, I found the answer to my problems - CygnusEd or CED for short.

CED

This editor is not a freeware or shareware product but is released into the Public Domain as a demonstration package for us all to play with and evaluate at our leisure. The 'demo' version is identical to the full function commercial version of CED except for one small limitation; you cannot save files larger than 3 Kbytes. Files of any size can be read in and manipulated in any way to investigate all the editing facilities provided, you simply cannot save to disk any file above that set limit. From this 'demo' version I was able to determine that CED would meet ALL my previously outlined requirements and more, this is an excellent way to release a product.

CED also supports overscan if you wish to run with larger windows than are available on the standard Workbench screen. It can be set as 'memory resident' although not in the standard AmigaDos sense, to become active from a simple keyboard command. It will work with lines of text up to 1024 characters long and is limited in file size only by how much free memory space you have on your system (even this limit is currently under revision). A maximum of 10 windows can be open in CED at any one time and these can be in the same file or different files if required. There are so many other features that I can only suggest that you get the 'demo' version and try it for yourself. The disk also contains 'ordering' information which I have included at the end of the article for anyone interested.

What else did I like? The extensive on-line help, the best file requester I've seen on any Amiga product, fast text searching, very fast text scrolling and, yes of course - the price - $30 US, to which I added an extra $5 to cover return airmail postage. The package was delivered within 4 weeks and came complete with a 20 page well laid out and helpful users manual.

I have received excellent support for CED, including a letter from CygnusSoft warning me that some of their distribution disks had been shipped to users with a virus on the boot tracks. They also described the effects of the virus, how to check for its existence and how to remove it from their disk and any of mine that may have been affected (infected?). Fortunately I had not booted to their disk, so my system remains clean.

I have found only one small bug. I sent a report for which I promptly received a reply acknowledging the problem and informing me that it had been fixed in the next version about to be released. Apparently this is a fairly major new release which is available to existing users for an update fee of $10. Their general update policy is that you send them as much money as you wish ($10, $20, $50?) and nominate how frequently you wish to receive updates at $10 a time.

CygnusEd is available from:-

CygnusSoft Software,
P.O. Box 363,
1215 Davie St,
Vancouver, BC,
CANADA V6E 1N4,

Phone: (604) 688-1085

==

Animation From C

by Alan Kent

In this article I thought I would pass on some experiences I have had trying to use the animation libraries in the Amiga from a C program. The kind of animation I am talking about here is not the high quality ray-tracing pictures which can be created with programs like Animate-3D, or creating animations using other packages such as Aegis Animator; instead, I wanted to try and write my own game (which I never got around to doing). I will also assume the background on which I want to do the animations is static.

The first step was to try and understand the primitives. Sprites on the Amiga are implemented in the special graphics chips (the copper and blitter etc). However, sprites can only have three colors each, so I decided against using them. The next thing I came across were Bobs (Blitter OBjects). These were similar in concept to sprites, except that to draw a Bob, the background is overwritten. To move a Bob while retaining the background involves saving a copy of the background where the Bob is going to be drawn so that it can be restored later. To draw a Bob then involves saving the background, then writing the Bob. To move the Bob involves copying the saved background back, changing the co-ordinates of the Bob, saving the background at the new location and drawing the Bob. Things get a little trickier when you have multiple Bobs on the screen and they overlap. Luckily the Amiga library routines take care of all this for you.

To do smooth animation however, you really need to use what is called "double buffering". The problem is when moving a Bob, the user will see a flicker as the old Bob image is removed and the new one is

drawn. To overcome this, the strategy is to create two background screens with the same background picture on them. One screen is displayed while the other is being updated. When all the Bobs have been moved, you flip the two screens (which can be done very quickly without any flicker). This of course takes up much more memory. A 32 color 320x200 pixel screen requires 40K bytes of chip memory - so 80K bytes if you want double buffering. All the images for each Bob must also be stored in chip memory, along with a buffer to save the background (two buffers if using double buffering). So, 10 little men (32 colors, 16 pixels wide, 32 pixels high, two backup buffers) would require about 10K bytes. Not too bad, but the more Bobs you want, and the bigger you want them, the more chip memory you use up.

Reading further in the manuals, I discovered this higher level function called Animate() which allowed you to design quite complex animation objects (called AnimObs) which could be built out of Bobs. The idea was that if you wanted a man to walk across the screen with his arms swinging, you could define a set of pictures for the different positions of his arms, a set of pictures for the different positions of his legs and say a fixed body and head picture. Each set of pictures were linked together using a structure called AnimComp (animation component). Oh, one thing they don't say in the manual is that each AnimComp sequence must have at least three Bobs in it or else it gets into a real knot. The rate to cycle through these images can be defined so the arms could swing faster than the legs if desired. One thing that confused me for a bit was an example in the manual where they had a series of four pictures and two pictures were the same (eg: arm forward, arm middle, arm back and arm middle). What they were doing was making multiple Bobs point to the same image in memory while having separate buffers for each Bob to save the screen background. Another advantage of AnimObs was that you could define velocities and accelerations as well as simple coordinates. These values are all stored as fractions (contrary to one manual which said the AnimOb coordinates were pixel co-ordinates).

This seemed all well and great. The system allowed you to develop quite complex objects which could move and the system handled most of the work for you. Then I tried using it all in a real application.

If you want to write a game, I advise forgetting the Animate() function completely for a number of reasons. First, having Animate() perform the movements using the acceleration and velocity information is generally too inflexible. You want to force objects to move around the screen where you tell them to go. Second, having multiple Bobs per object slowed the whole system down due to having more Bobs on the screen and as overlapping Bobs require more work to manage. The third problem was that it was very difficult to work out which Bobs were currently on the screen. This is important if you want to do Bob collision detection (to see if a missile hits a space ship for instance). It is also a problem if you want to force one man to be in front of another (or even to define that an arm should appear in front of the body) as you have to define this at the Bob level using Before and After pointers. Changing this information dynamically (for example having two people walking around each other) as the program runs is not fun. The real killer in my opinion however was the sheer amount of memory required. Every single image must be assigned a Bob structure with backup buffers. AnimComp structures must then be set up to group the Bobs into a sequence (such as a swinging arm). An AnimOb structure must then be set up to hold details about all the AnimComp structures for the object. Having two identical men allowed the pictures to be shared, but not the backup buffers, Bob structures, AnimComp structures or the AnimOb structures. Any slightly complex scene I tried to set up ran out of memory. There were other problems too. In summary, I think the Animate() function is next to useless as it is too restricted in the way it can be used and too greedy for memory.

The solution I used in the end, which works quite well and is fairly memory efficient, goes back to using Bobs. Rather than defining the swinging arms and things separately however, I drew a series of complete pictures of a man walking along. This allowed the man's body to bob up and down or do whatever it liked while walking along. It was also much easier than trying to synchronize separate series of images too. To make the man walk, I defined a single bob with two backup buffers, then swapped the image it pointed to for each frame. Each new man I wanted to create only required two extra backup buffers (for double buffering mode). It all ended up much faster, easier to control and more memory efficient. Making sure the one Bob is in front of another Bob is still a little tricky, but not too bad.

This took me about a year to work out in my free time, which was mainly spent trying to get the Animate() function to do what I wanted.

CAD for the AMIGA

APAL IS FULLY FEATURED 2-DIMENSIONAL CAD

No compromise Computer Aided Drafting is now available. APAL is Computer Aided Drafting with all the features you need. Proven user Interface based on leading minicomputer CAD system. Fast and accurate for high quality Plotter or Printer output. Local support is assured.
Call or write for further details and nearest dealer.
Version 2.2 now available
PRICE $250.00

BASFORD SYSTEMS

25 Neale St., Bendigo, Victoria, 3550
Telephone: (054) 43 3349

MADE IN AUSTRALIA

MIDI - Bringing Machines Together

by David Galea

The music industry has long recognised the need for the implementation of some kind of universal interface to allow all keyboards to communicate freely, and MIDI is an attempt to provide it. It stands for Musical Instrument Digital Interface, and has opened a new era in music for even the beginner.

After lengthy consultations with most of the worlds major keyboard manufacturers, the format has been drawn up by Sequential Circuits, maker of the famous Prophet synthesisers, in close co-operation with Roland. There are now very few manufacturers who don't incorporate MIDI into most of their range.

The only physical feature of a MIDI instrument to give it away is the inclusion of a couple of 5-pin DIN sockets (as on your stereo) on its rear panel, marked "MIDI IN" and "MIDI OUT", and these will allow two-way communication between a pair of instruments. Some models will have a third socket or "port" marked "MIDI THRU" which allows the connection of more than one instrument in a chain configuration, with the first one in the line being the master, and controlling the others. This is utilised to a great extent by many modern bands who have a single keyboard held in a guitar-like fashion, which seems to have a very filling sound, but what you don't see is the other synths backstage pulling their weight.

The system is by no means limited to synthesisers, but it's also equally useful as applied to electronic pianos, other electronic keyboards, sequencers and drum machines.

One of the really exciting areas opened up by MIDI is that of being able to control many instruments from your personal computer (Amiga). This has allowed relatively sophisticated recordings to be realised within the limited resources of a small home recording setup, by programming the various parts to be played simultaneously by the Amiga.

In the past this kind of interfacing has been solely the domain of the electronically knowledgeable, or the wealthy professional, who could afford the very upmarket instruments, but now since the advent of the Amiga, a new era has dawned and with plenty of software support, its future looks bright ahead.

==

FFS and Janus - A Marriage Not Made In Heaven

by Eric Salter

When Workbench 1.3 becomes available, people with hard-disk drives will benefit from the new file system - "Fast File System" or FFS for short. This new file system is an alternative to the current file system that we all have loaded on our 3.5" disks. Its big advantage is the speed-up that will be gained - from 5 to 40 times the speed depending upon the operation. As of this writing however, people with Bridgeboards and IBM ST-506 hard drives in which part of the drive is partitioned for AmigaDOS, cannot use FFS. We will examine why and pose a solution.

The file system is the process that allows us to read and write data as files. It is one of many tasks that are running on the Amiga. This process, like all processes, is part of an amorphous thing called AmigaDOS. AmigaDOS itself is a library of routines + a global data area + the "Global Vectors" of processes + all the processes running - these include the file system, the console (CON), PAR, SER, RAM etc. AmigaDOS works like this - if a process wants to send data to a file on a disk or the printer or whatever, it sends a message to another process that is handling that thing. This process is known as a handler and is the basis of AmigaDOS. A handler responds to a limited number of requests that are given to it in these messages. The messages are known as "DOS packets" and are built on the Exec message system and use Exec to pass them back and forward between processes. The dos library comes into the picture when we want to create these dos packets. If a program wants to Open() or Close() a file, it will call one of the functions in the dos library - a process we can't go into here. This function will then create the required dos packet and send it to the relevant handler process via the message system. This is totally transparent to the casual programmer, but can be taken to its logical conclusion - it is possible to ignore the dos library altogether! (well sort of). If you know what type of packet you want to send and where it has to go, you can send it to the process directly without having to go through the dos library routine. This method offers less overhead in the long run.

The handler process, in turn, translates these dos packet requests into the Exec I/O functions that control the device which controls the hardware. One dos packet request can involve many operations at the Exec device level. The file system process is just such a handler. The file system knows about directories and files and drives with names like DF0: etc. If it receives a packet request saying "Open file 'x' on disk 'y' in directory 'z'", it knows how to get at that information by looking at special data structures that it has created on a disk. For a 3.5" floppy disk, these data structures where written when the disk was formatted. The file system requests that the trackdisk device, that is the device controlling the actual hardware of the disk drive, bring in a particular track and sector. This sector will have information that the file system can use to look up where the file is on the disk. It then asks the trackdisk device to bring in further sectors and tracks. The trackdisk device does not have to know anything about files and directories, only how to get sectors of data off the disk. What is stored in those sectors is the sole responsibility of the file system.

FFS is a new version of this file system. What makes it different from the old one is in the way it organizes these data structures on the disk. Currently, data in a file is stored in sectors along with additional file system data structures. This means that when a sector is read in from the trackdisk device, the data has to be extracted from the file system data before it can be used. This is valuable processor time spent moving data around that it really can't afford to do. This is why most hard disks were slower than their IBM counterparts in data transfer rates - the slow-down was in the file system itself. In FFS, a sector of data is a pure sector of data without file system junk. The practical upshot of this is that a sector can be DMA'ed into RAM where

it is supposed to go rather than into temporary buffers where it is translated first. Another feature of FFS is that sectors of one file tend to be allocated to contiguous physical sectors on the disk. This means that instead of loading one sector at a time, an entire chunk of file can be read in one go, speeding up I/O markedly. The final feature of FFS is in the way sectors are buffered. In the old file system, one could allocate a number of buffers to be used as sector caches. Any sectors that were in these caches did not have to be read in from disk - or written out to disk until convenient. Under FFS, these buffers or caches are used to cache directory data - thus speeding up one of the most hated features of AmigaDOS - the slooow directory listing. Files whose names are known are found blindingly fast. The final feature is the ability to have volumes with more than 51 Megabytes size. The sky's the limit now.

Now the problem. FFS cannot currently be mounted on a Janus disk drive. The reason is that you cannot format the disk for FFS. Under 1.3, the new format command allows you to format a drive under FFS or the old file system. Format expects to use the same device commands as the trackdisk device. Unfortunately, the jdisk device, which is the device to control the AmigaDOS partition on the Bridgeboard, does not support the FORMAT command, in fact, Commodore supply their very own command to format a jdisk drive - DPFormat. The Amiga gives a little grunt from the hard disk and then dies if you try to format the jdisk device (JH0:) with the "format" command.

The solution - the disk must be edited by hand to make the correct image for FFS to recognize this as a disk formatted under FFS. You need three things.

1. A correctly constructed entry in the mountlist for the drive JH0: - something like this:

```
JH0:        Device = jdisk.device
            FileSystem = l:FastFileSystem
            Unit = 0
            Flags = 0
            Surfaces = 4
            BlocksPerTrack = 17
            Reserved = 1
            Interleave = 0
            LowCyl = 0 ; HighCyl = 624
            Buffers = 30
            GlobVec = 1
            BufMemType = 3
#
```

   Of course this is just an example and your own disk drive characteristics would have to be considered when determining the number of tracks and surfaces etc.

2. Using a disk editor that allows you to edit tracks and sectors of any AmigaDOS device, mark the disk as being an FFS disk. Track 0, Sector 0, Surface 0 must have the following 4 bytes as the first 4 bytes of the sector:

   44 4F 53 01

   which are the ASCII characters: "DOS(01)". Under 1.2 and 1.1, this used to be: 44 4F 53 00 or

MEGADISC

for the *AMIGA*
INFORMATION &
ENTERTAINMENT

7 Disk-Magazine Issues Available

All 7 issues of the MEGADISC Disk-Magazine contain the information you need to make the most of your Amiga. Designed to be easy to use, all you have to know is how to click on the mouse buttons. Every issue is packed with information - articles, tutorials, reviews, free utilities, illustrations, where and how to get the hardware and software you need, along with the latest updates locally and overseas. With MEGADISC, you learn as you use, and you'll like learning. MEGADISC explains from the beginning, and entertains to the end. ** WATCH FOR OUR *MEGADOS* AMIGADOS-ON-DISK MANUAL AVAILABLE SOON, WITH SPECIAL OFFER TO CURRENT SUBSCRIBERS! **

SPECIAL OFFER!

Order ALL 7 CURRENT MEGADISCS now for $99 (a saving of $40 over normal prices), and ask for 4 FREE disks from our Public Domain Library. If you don't know which free disks you want, ask for our Catalogue-on-disk and choose three more later.

300 PUBLIC DOMAIN DISKS FOR $5.50 EACH
All these disks are fully described on the Catalogue-on-Disk.
GET ANY 10 PUBLIC DOMAIN DISKS FOR $45
GET OUR "GAMES PACK" - 5 DISKS FULL OF GAMES - $25
WE DO NOT CHARGE FOR POST AND PACKAGING

I enclose a Cheque/Money Order for .... or please charge my
BankCard/MasterCard No. ....Expiry..................
Please send me: Catalogue-on-Disk ($5.50): ....
ANY 6 issues of MEGADISC for $90 (please specify): ....
ANY 3 issues of MEGADISC for $50 (please specify): ....
ALL 7 issues of MEGADISC for $99: ....
The 4 free PD disks I want (2 in the case of a 3-issue sub) are:
OR please send your Catalogue-on-disk now and I will choose the remaining disks later.
GAMES-PACK for $25:.......................
Signature:..
Name: ....
Address:..
Postcode: ....Telephone:............................ Date: ....

"DOS(00)". The reserved field in the mountlist stops the file system writing over this DOS flag by reserving the first sector.

3. Creating the Root directory. In the DOS manual, the file system structure is discussed. The Root block for both old and new file systems is essentially the same. Fooling FFS into thinking this is an FFS root block involves making a false root block with all hash table entries set to ZERO and marking the bitmap true flag as false i.e., offset 78 (the 78th longword in the sector) set to ZERO. The Root block is located in the middle of the disk between the high and low cylinders. Now, mount the disk with the command:

```
    mount jh0:
```

not as before with DjMount. This should now tell the system to load FFS from the L: directory and use this to access jh0:. The rootblock has been set up such that the restart validation process will see the bitmap true flag as being false and will set about re-validating the disk, thus creating the bitmap sectors and sparing out those sectors owned by the system - the root block, the bitmaps sectors and the one reserved sector. You should now have a validated hard disk running FFS.

I hope this will help developers play with FFS on the Janus system and act as a stop-gap measure until Commodore get their act together and re-write the PC Emulation software to run properly.

===

Deluxe Laser Painting
Computing for Fun and Profit
by Neal Glover

"The Amiga is a great graphics work station" (among other things). That's the reason why I bought it, and I have certainly found that statement to be true. My line of work involves graphic art, mainly for real estate advertising (sketching houses and so on), and while most drawings still need to be done using good old pen and ink, the Amiga is becoming an increasingly invaluable tool of my trade. D-Paint 2 PAL and Digi-View PAL are my favourite packages as they are such a versatile combination. From what I've seen of Photon Paint, it looks like it also will be added to the favourite package list.

Much of the work I produce on the Amiga starts life as a digitised photo, artwork or sketch. (The house on the cover of Nov '87 Workbench, the F-16 on the March '88 cover and the church on the front of the April '88 issue are such examples. The process, then, is to simply trace over the digitised image using D-Paint 2 (or similar), and then remove the excess shades using the palette control. I say "simply" about tracing, but it's not really so simple. Perspective is difficult to get exactly right, and most photographs distort the perspective to an unacceptable degree for drawing purposes. All this has to be manually corrected, and is a long process, especially drawing lots of detail in high-res. D-Paint's "Perspective" function will only put perspective onto a 2D object by rotating is about 3 axes and adjusting the spectator point. True 3D perspective requires other methods, some of which can be used in conjunction with D-Paint's "Perspective" function. (eg Pasting a label on the side of a box drawn in perspective. Indeed, the label may form one entire side of the box, and a 3D object can be constructed in this way. But for complex drawings - forget it!).

Hopefully, somewhere in this edition appears a drawing of the "Colonial Hotel", which had to be drawn from scratch on the Amiga - no Digi-View here! (The "Colonial Hotel" drawing is copyright as indicated, so please don't go re-copying it!). Although the final print is in eight shades, the initial line drawing was done using only four colours, in hi-res. A line drawing has only two shades - black and white, so why four colours rather than two? Well, it so happens that the other two spare colours are very useful for drawing perspective guide lines to the imaginary vanishing points that all perspective drawings (ray tracing and other number crunching methods aside) require. As they are only guides, they can be removed later by a palette adjustment, leaving the drawing in black and white. A line drawing in which proper perspective is not critical, or doesn't exist (plans and elevations), needs only two colours. The fewer colours the better if you are using hi-res. Large brush manipulations can be a problem even with 2.5 megs, as chip RAM sometimes doesn't go far enough. (Roll on Fat Agnus!).

As all my drawings are output to a laser printer, they are all done in beautiful living B/W - at least until somebody comes up with a cheap colours laser! Grey scales are easy to set up in D-Paint 2 using the "spread" function of the palette. Of course, digitising in B/W does it automatically. D-Paint 2 and a laser are quite a good combination for producing leaflets and handbills, although bit-mapping restricts the quality somewhat (Postscript Desk Top Publishing it aint). Again, anything like this is best done in hi-res with as many colours as you need or your RAM can handle. Hi-res minimises the bit-mapped effect. The bigger the page the better. An A4 page is best set up with a D-Paint page size of 640 x 904. 2.5 megs is adequate for such a page size in 16 colours, although brush manipulation may be a bit tight. This gives good resolution on the print without the need to scroll sideways on the screen. The bit-mapped effect can be further reduced by using hi-res fonts such as the excellent "Pro Video Fonts" by JDK. "Smoothing" and "Anti Alias" functions can also be useful, but as many shades as possible must be used to get the full benefit from these functions. Up until recently, printing out Deluxe Paintings on my laser has been a bit of a disappointment, in that the Amiga wouldn't drive it at its full 300 DPI. That's all changed thanks to the new preferences and printer.device soon to be seen in AmigaDOS V1.3. I'm running a Ricoh PC Laser 6000 with a HP_LaserJet_PLUS card and 1.5 meg RAM (for full A4 graphics). The results are really pleasing (with 300 DPI!). Hopefully by the time you read this, Ricoh will have brought out a Postscript card for the 6000. (Professional Page here we come! - looks like being an excellent program).

I recently hired a Mac SE to check it out with some DTP/graphic programs (Pagemaker, Mac Paint/Draw etc). The only program that impressed me was Adobe Illustrator. I wouldn't be disappointed to see that or something similar for the Amiga. However, after

recovering from what Pink Floyd would call a "Momentary Lapse of Reason", I returned the Mac with an Amigan smirk on my face. Keep trying, guys! Seriously now, the Mac just didn't seem to cut it with Amy's graphics. However, the Mac's list of available peripherals are impressive. Digi-View is a fine program, but a scanner (preferably a flat bed) would be nice as well as a few other toys to keep graphics-gurus happy. Hopefully with good DTP packages such as Professional Page appearing, local typesetting bureaus will be connecting Amigas up to their Linotronic typesetters. I'm planning to go 300 DPI Postscript, and in addition to me normal work, hopefully will be able to offer AUG members a Postscript printing service.

That's enough from me now - happy number crunching.

===

Uninvited Review
by Glen Sheppard

Uninvited is a text adventure in which your car has crashed and you go to a nearby house (which just happens to be haunted) to use a phone. Inside this house, there are several rooms, some halls and towers. There is supposedly a telephone, kitchen and several other rooms which I have never been able to find. Secret passages are also around the house, but who knows where they are?

I have noticed that in magazines lately, Uninvited never seems to be printed on the software lists. Whether this is due to lack of sales, I don't know. I don't know of anyone else with a copy of Uninvited on the Amiga, let alone any other computer. I would love to hear from anyone else with the game.

Overall, I'd rate this game as quite good. Graphics are good, and sound isn't fantastic but adds some life to the adventure. All movement is through the mouse, so you can sit back in your chair and not have to type. A "save as" mode allows you to come back to your current position, but you must save it onto a separate disk like Flight Simulator II.

For me, Uninvited would be worth the money if I could find the things I mentioned above. If anyone does know where these passages are write in and tell me!

===

Test Drive Review
by Glen Sheppard

Does driving a car along a mountain road sound fun in your old car? Why don't you trade it in on a Ferrari or a Lamborghini? Your neighbours will suddenly become your best friends. Too cold to go and look at all of these cars, eh? Rubbish, all you have to do is pull backwards or forwards on your joystick to choose the car that's right for you.

The object of the game is to drive a Ferrari, Lamborghini, Porsche, Chevy or Lotus along a twisty mountain road to the next petrol station in the shortest time possible. To succeed in this game you have to dodge cars, trucks, potholes, rocks, the cliff's edge, the police and the odd dropping from a passing bird. The best score I have achieved between one petrol station is a huge 69900 with a total score of 121880 in the Ferrari.

Now for a hint. I find that when I am driving too fast and look like crashing in a sharp corner, holding down the joystick button the car will take the corner itself. But beware, because when you release the button, your tyres will head the same way they were before you pressed the button, so be careful.

One complaint I do have with Test Drive is that if you have sped up when the radar detector goes off, the police can still overtake you even if you reach 150+ mph while their car is in your rear-view mirror. A picture on the back of the box shows a part of scenery that I have never seen during the game, perhaps this is contained on the mysterious scenery disk that is rumored to be in the making.

I enjoy this game for its great graphics although there is no variation in scenery. Sound effects are very good with engine, tyre, radar and theme music; game control is with a joystick.

I would recommend Test Drive to anyone who likes to drive very fast, it's safe on your Amiga. Test Drive is a relatively cheaply priced game at only $49.95, considering how much detail is in the game.

An interesting point is that the music from this game can be used in Deluxe Music Construction Set, and the title screens can be viewed on Digi-Paint and transferred to DPaint 2 if you wish to change colours or something.

===

Silent Service
by Glenn Sheppard

Silent Service is a game based around a World War 2 United States Navy submarine and its confrontations with the Japanese Navy, with you in charge of the sub.

Once the game loads, you have a choice of torpedo/gun practice, Convoy actions or a War patrol, which can last 57 days. After your decision, you choose difficulty level, rank and features of the game which include accuracy of destroyers, when you can repair your vessel etc).

Your mission is to sink as many enemy ships as you can without getting sunk yourself. Enemy ships include destroyers, cruisers, kaibookans, aircraft carriers, troop ships, tankers, and cargo ships. Your submarine is armed with torpedoes, both front and rear (which have a range of about 5000 yards) and a deck gun (with a range of about 3000-5000 yards, depending on sea conditions). Your mission starts at one of three bases, and you navigate the sub around with the mouse, joystick or cursor keys.

This game has good graphics (periscope views, maps, gauges, the bridge and ships on fire), realistic sound (sirens, torpedo launching, sonar, gunshots and explosions), and easy control over your movements. On the negative side, changing between the screens to a different room in the sub is slow.

Games SIG Report

by Luke Devlin

A great turnout at the AGM July meeting. Congratulations to all those who were voted in, and thanks to all those who nominated for any positions.

At the July edition of the Games SIG, we showed several new games just right for the Amiga's capabilities. These were Bubble Bobble, Barbarian (the other version with two sword fighters and the gremlin who kicks the decapitated head along like a soccer ball), Space Racer (the speed biker game), Pink Panther (all 20 seconds of it), Kickstart II, Joe Blade, XR 35 (all 30 seconds of it), Roadwars, Interceptor and B.M.X simulator. We had a lot of fun with some of the harder games like Pink Panther, as I can only last for 20 seconds. XR 35 is another hard game, and I challenged other members to get more than 30 seconds into the game without getting killed. Anthony almost lasted but was killed right on the buzzer. We found an expert at Kickstart II to show us what it was all about, and Robert showed us BMX Simulator and all those sneaky short cuts. Another keen games fan killed me several times on roadwars to the satisfaction of the crowd. Also we helped people with answers to their questions on games, parts that they couldn't get through, where you should purchase, the best price, is it better to purchase from the states and so on.

Don't forget to bring along your latest games to the next Games SIG, hoping to see YOU there!!!

Games SIG organizers are Luke Devlin, Anthony Woods, Robert Nunn.

==

SMAUG Report

by Doug Myers

No! SMAUG is not the name of an elusive dragon invented by J.R.R.Tolkien. It is the acronym for the Sub-group Music AUG, and it was the first interest group established within the Amiga Users Group.

The first SMAUG meeting was held on the 8th of February 1987 at Roland Seidel's house in Box Hill, and Roland was elected as the first co-ordinator of the group. Roland's house was about three minutes drive from the Burwood Teachers College where the AUG Meetings were held and so it became a pleasant duty for the members of this group to drop into Roland's before the main meeting, quaff down a champagne or two and watch a demo or listen to some music on Roland's setup. The group went from strength to strength until our committee heard of its success. An urgent meeting was called and it was decided to change the venue for the AUG meeting to Monash. "That should fix those boozing musos" one of the committee members was heard to say. And it nearly did!

SMAUG was allocated a small room in a circular building and the members had to carry gear from everywhere for the demos. It was getting to be more difficult. The booze stopped flowing and the musos began to look a little limper. Fewer faces were seen in the Rotunda. The demos began to sound the same. The feet stopped tapping. Had the committee won? Was it Sunday or Saturday? Who cared anymore?

At 4.30 on the afternoon of the Annual General, a few old and new SMAUG members congregated without Amigas, midis, synths or booze to contemplate the situation. The news had come that Roland had resigned. Would SMAUG go the way of other interest groups when their co-ordinator resigned? All agreed that Roland had done a very good job, but he wasn't indispensable. We were cheered to hear that he would still write music articles for Workbench and be a part of SMAUG, and then we contemplated the Future. How could we get back to enjoying the music and foiling the Committee at the same time?

There were several problems. The first was the specialised gear needed to demonstrate music and the second was the different levels of expertise amongst the members of SMAUG. It was decided that a new co-ordinator was needed but the best offer we could get from the floor of the house was for a three month period only. The idea was put forward that we might use member's homes for demonstrations and they could provide the gear and demonstrate their setups. We also had the offer of the use of a professional computer music centre on occasions. SMAUG was seen as a special interest group which, like the video people, had to meet in specialised surroundings and probably outside the regular meeting times.

It was decided that the next meeting of SMAUG would be held at the Rotunda as usual after the August meeting. The meeting would contain some of the latest examples of public domain software available in the music area that are guaranteed to keep the feet tapping. If time permitted, there would be an introduction to Larry Spiegal's Music Mouse. The main purpose of the meeting would be to decide on the future venues and times for SMAUG meetings and to elect a co-ordinator and perhaps a committee to help organise the activities of the group.

This is an appeal to all the members of SMAUG to attend and bring back the beat which has been getting slower and softer. In this edition of "Workbench" Roland has submitted "The Idiot's Guide to SoundScape", the first of a series which we must produce called "The Idiots Guide To........". Please think of ideas which will help to make SMAUG more interesting for you and bring them to the meeting.

The co-ordinator of the August meeting will be Neil Rutledge whose address is 13 Marshall Ave. Highett 3190, phone 597 0928. Give him a ring if you have some ideas which won't wait.

[Editor's Note: Unfortunately, Roland's article hasn't turned up in time - maybe it'll appear next issue.]

==

AUG Business SIG

by Neville Sleep

The July meeting of the AUG Business SIG was not attended by its originator Chris Noble, however there was enough interest for such a SIG to develop into a sizeable group.

The format of the group at its first meeting was as a general discussion on the Pros and Cons of general business software, with inexperienced members such as myself gaining valuable clues on the operation of various programs and finding some features we previously never knew existed. The format for future meetings may change as the viewpoints and preferences of members become apparent.

At the August meeting of the AUG Business SIG, John Barlow will have an Amiga available to demo "Maxiplan" and will be available to answer any of your questions. Demonstrations of other programs and devices have been arranged for future meetings and should ensure interest in the SIG well into the future.

==

The history and future of the C SIG (Beginners)

by Mal Woods, C Sig Co-ordinator

Many meetings ago at Burwood Teacher's College, SIGs were announced. It sounded like a good idea to me. People with like interests meeting separately to discuss their chosen topics. Out of the SIGs that were available, I decided to attend the Graphics SIG as it was the topic that interested me the most. So accompanied by some close friends, we sat in the back row in an attempt to NOT volunteer for anything.

During the discussion it became obvious that there were a couple of different opinions as to what the group should be doing. Some wanted to know how the graphics routines worked and how to use them while others wanted to use packages which already exist to produce pictures. I was of the first group. Those in this group decided that to be able to call the graphics routines they would have to have at least a rudimentary knowledge of C and it was decided to form a C SIG to focus on this point. But who to control such a body....

Bob Scarfe was attending to help us get started and when he asked for a show of hands as to who would attend a C SIG, I and all my friends put obliged. Bob took out a piece of paper (as though to start taking names) and pointed at me (the person closest to the left wall) and asked for my name, which I of course gave. Instead of asking the next person for their name Bob announced that I would be the coordinator for the new C SIG. I was a little taken aback. I knew little C and as such thought I would not be qualified. I also ran another computer club and already had a lot of my free time taken up with the organisation of this club and the production of the newsletter. But still I thought I would give it a try and after the meeting I talked to Bob and he assured me he would do anything to help me out. My friends also assured me of their assistance.

Well a month went past and the next meeting came around and I cautiously took my place in front of a reasonably large crowd to try and organise things. They were a reasonably friendly bunch (nothing was thrown), and in discussions with them it was found that a majority of them didn't know C and would like to learn it. "Fine", I said, "It just so happens that I have a public domain C tutorial which I have started to work through which we could all do together." There was a general agreement amongst the beginners and a slight moan from the gurus in the front row.

We then proceeded (with the help of the gurus), to write a small C routine on the board concerning the state of a wombat crossing the road. This choice of topic was not only educational, but amusing as well. I left the meeting with high spirits of what was to come.

I don't know what I did wrong at that meeting but the number that attended the next one was significantly fewer. We struggled on regardless. Our store of gurus had been reduced and my preparations were not the best, but we successfully started the training. Copies of the text were handed out and examples were written from my originals on the black board by Peter Jetson who also sat in the front row and corrected all the mistakes I made. He was a great help at that meeting and many to follow.

At the later meetings (at our new location), I got a bit better organised. I had prepared overhead projector slides with the examples on them. I didn't want to have handouts with the examples on them as I thought that people would learn them a lot better if they wrote them down. Things went well, but attendance continued to drop off. Some people at the earlier meetings contacted me and got a copy of the tutorial for themselves (my friends among them). After that had that they had no further use for my SIG. I was killing the SIG myself. Peter found that with the extra responsibility associated with the club he could no longer attend. Eric Salter filled in nicely for a time until he left to start up his own SIG (Advanced C SIG).

At one meeting I had all my slides ready but no one could find the overhead projector, so we had to revert back to the blackboard method. After this, I decided that photocopying the examples would significantly hasten the learning process. It certainly did. We went through a whole chapter and a half in one sitting (a record that has not been broken since). We also missed a meeting due to a demonstration taking place in the room we were supposed to be in.

At the last meeting Bob met with the SIG leaders after the main meeting to express his concern with the way the SIGs were run. Some SIG leaders didn't organise replacement leaders when they were unable to attend, others just stopped turning up and their SIGs fell apart. He wanted us to try and run the SIGs like the club was run (with a committee and an election of our own). My SIG does not really fall into the mold defined by Bob. When I went to the meeting there was a lowly four people there (a couple turned up later), but if a committee was elected then everybody would have a position, so I politely ignored Bob's suggestions and continued on as normal.

Bob didn't let me off so easily. He walked in and all those that attended discussed with him the future of this SIG. Though we thought we could not be likened to the Beginners SIG (who have 4 lessons which they repeat over and over again to cater for new beginners who enter the club), we had a lot more to go through (we're only up to Chapter 7 now). We came to an agreement that to try to appeal to others in the club, we would turn things around a bit. Instead of going through the tutorial in the SIG, those who wanted to learn would take home the notes and teach themselves between meetings. At the meetings themselves we would help people with problems they had with their (simple) C programs (with priority given to the examples in the tutorial) and hand out the next chapter to be studied at home. I decided to bring my Amiga along so that compilations could be done there and we could try to determine the answers to any problems raised.

What all this above blurb is getting to is that I need your help in two ways. If you are a beginner with problems in a C program you are working on then please bring it along on a disk and we will attempt to solve it for you. If you are an expert please come along and help us solve these problems. I still don't know a lot of C, but I'm getting there. With your help, we can provide a useful service for the members of the club. It may be that we will change our name to the Beginner's C Problem Solving SIG. Who knows. Come along and have a say. Please.

==

Advanced Graphics SIG

Meeting 3

by Geoff Holden, the Reluctant Convenor

Well. It seems that the Holden Show and Tell Traveling Circus, also laughingly known as the Advanced Graphics SIG is about to change into something deep and meaningful and useful. I will pause until the cheering dies down. This came about because (a) Bob Scarfe, our beloved President, descended upon us like a fiery angel, sword in hand, and demanded that we organise, clarify and generally clean up our act, (b) I have some additional work commitments that you wouldn't believe, and as you wouldn't believe, I won't tell you about them and (c) it was getting so boring, I'd stopped laughing at my own jokes. So.... a Gang of Five has been appointed to make Advanced Graphics the Colour of the Month and generally laugh at my jokes.

The innocents involved are: Dick Bartholomew, Joe Santamaria, John Barlow, Dan Davies and Your Humble Servant. The idea is that we should meet, in between AUG meetings and plan delights for ourselves, and - with any luck - you lot out there. The three ring circus sounds like a start...

I am not publishing the phone numbers of the Gang of Five - yet. This is a weapon I will reserve for disobedience (not helping, not laughing at my jokes, etc). I will probably continue to stand up front and gibber, since no-one else thundered forward to volunteer for that mouth-drying duty.

Meanwhile, back in the real world, at the July meeting, Con demonstrated some more of the DMWRender public domain ray tracing program. The ability to add a variety of textures to the shape primitives was the most interesting feature of the program - if only it didn't have such a ghastly input system for data! Mind you, if it was as uniformly good as its best features, I would weep all over my wallet for having bought Sculpt and Animate 3Ds for more money than I'd ever tell my children. Con will probably continue his demonstration (a sort of Dance of Death with the Guru) at the August meeting.

So, if you raytrace in your sleep or practise solid constructive geometry on your accountant, then octree your way along to our new, improved, non-singing and absolutely dance free Special Interest Group at the August meeting. We'll see both of you there.

==

Introduction to the "C" Programming Language Part 4

by Eric Salter

Last episode, we looked at the tools of the "C" programming environment. This time we will look at the basics of the language itself. This is not meant to be an extensive tutorial on "C", but hopefully it covers the main features and structure of a "C" program. Ideally, it should facilitate understanding of the copious amount of Amiga documentation written with the "C" programmer in mind.

A C program is simply a text file that you create with a text editor or word processor. You cannot run it directly because it has to be compiled. A C compiler must read your text file and convert each C statement into the binary code that your particular CPU understands and can execute directly. After this, your C program will run as a machine language program - fast, unlike BASIC, in which the BASIC interpreter must interpret what each statement is doing, every time the interpreter reads them. Another advantage is that a C program stands alone. It doesn't need any "run time" package to be in memory with it. A BASIC program needs the BASIC interpreter to make it do useful things and therefore C programs (usually) require less memory to achieve the same result as an interpreted program.

The C program consists of several sections which are part of every complete C program (but not necessarily part of every C source file):

```
Pre-processor directives, macros etc.
External Function and Variable declarations
Static Variable declarations

main() function
Function 1
Function 2
Function 3
.
.
.
Function n
```

Every complete "runnable" C program has a "main()" function. It is not necessary however, for all C code to have this function before it is compiled. Programs can be constructed from a number of separate "C" source files that have been compiled independently. Only one of these files need have the main() function defined. The necessity for a C program to have a "main()" function is due to the fact that at "link" time, when the executable code is created, the linker brings the code modules together such that your program will begin executing from your "main()" function. In fact, many of the statements in a C program are references to functions that are not defined in your program, but instead reside in libraries of functions or other code modules outside your program and are linked in to your code by the linker.

Pre-processor Directives

Any line in a "C" program with a "#" in the first column is not part of the C language per se. "Statements" here are called compiler pre-processor directives and belong to the compiler. They are never executed but are acted upon by the compiler when the "C" text file is read in. We will defer looking at pre-processor directives except for the more common ones.

#include - The include directive

The "#include" directive conceptually stops your program text being read in and causes the file listed after the "#include" to be brought in and inserted into your code at that position. The most common use for this directive is to bring in standard headers that contain definitions of system constants and functions related to file access or the operating system. These definitions, once read in, can be used as if they were part of the code. The "C" compiler will compile these header files along with the rest of the code in the file. The names of these files are system dependent but some fairly constant ones are:

```
#include "stdio.h"   - I/O macro definitions
#include "ctype.h"   - character testing Macros/type
                       defs
#include "math.h"    - on UNIX systems define special
                       math functions.
```

And some on the Amiga:

```
#include "exec/execbase.h" - Exec library structure
                             defs
#include "dos.h"           - AmigaDOS interface
                             structures
```

You will note that the name of the file you wish to bring in is enclosed in double quotes. This causes the compiler to search for the named file in a particular place, usually the current directory. If instead, we use angle brackets <> to enclose the name, this, depending upon your implementation, will cause the compiler to search for the file in the directory that contains standard header files - Quotes on the Amiga under lattice 3.10 searches the current directory first, then the "INC:" directory, whereas brackets search ONLY the INC: directory. On UNIX, the include directory is "/usr/include". Include files themselves may contain further "#include" directives and so on. This is called nesting and there is a limit, depending on the compiler, as to how many levels we can nest this directive.

#define

The #define directive allows us to make extensions to C by defining macros. A macro allows us to define a shorthand way of expressing constructs so that they convey more meaning as to what is actually done than would the construct itself. It also may save keystrokes! The simplest form of macro is substitution as with constant definitions:

```
#define MAXLINENUMBER  66
```

From now on, references to MAXLINENUMBER are equivalent to decimal 66. Not only does this convey more meaning than a raw "66", but it allows us to change the constant at one place in the text and have it changed throughout the program should we need to at a later date. The other use of macros is making simple from the complex:

```
#define  max(x,y)  ((x) > (y) ? (x) : (y))
```

Here, we have defined the macro max(x,y) with parameters. With each use of max(x,y) in our program, the compiler will substitute the code x > y ? x : y (the ternary operator). This makes the code, not only simpler, but more easily understandable. In one issue of BYTE magazine, there was an article describing a way to vandalize C by re-defining "{" as "begin" and "}" as "end", to make C look like Pascal code. Please, don't make C harder to read than it is. Note that a macro, such as the one above, is not a function call! It will be expanded into code, "in-line". Please be careful therefore, as expressions passed to macros may be evaluated twice. This may become the source of program bugs if you pass an expression with function calls or ++ or -- pre/post increment/decrement type operators (it will evaluate for each occurrence of the operator!).

Functions

Functions are part of all C programs. The main() function is also a function - the place where execution commences. Once a function is written and debugged, it doesn't have to be re-compiled, and we can think of it as a black box where what goes in and what comes out are known, but the mechanics of how it works become unimportant - we have added a level of abstraction, each level up being closer to the overall concept of what the program is doing. You could think of a program as an hierarchy of administration functions, each with the task of overseeing part of the job, with the workers at the grass roots doing the real work. The concept of functions that are units in themselves saves us re-inventing the wheel, and C programmers often develop their own library of functions that they have created over the years and use in their everyday code as if they were part of the language. The structure of a function is:

```
type function_name(argument_list)
argument declarations
{
     declarations and statements usually
     including a "return()"
}
```

For example, the max(x,y) we defined as a macro could also be written as a function:

```
int max(x, y)
int x,y;
{
     x > y ? return(x) : return(y);
}
```

Here we have defined our function called "max" to be a function which will return an integer. It has the formal parameters "x" and "y", which are defined as integers. The body of the function is the ternary operator as before, except that the statement "return()" is passing control and an integer value back to the calling function. We can see that a function can have multiple terminations. For purposes of explanation, the function could also have been written as follows:

```
    int max(x, y)
    int x, y;
    {
         if (x > y)
              return(x);
         else
              return(y);
    }
```

Note also that we have defined the two arguments "x" and "y". We must do this as these are not the same variables that the function will be called with, even though they do, of course, have the same value. They are local to the function. Whatever we do to these variables cannot affect the real variables in the calling routine. We will look at this again when we talk about the scope of variables as being one of the attributes of a variable.

Call by Value vs Call by Reference

In the above example when we call the function, probably from main(), as in:

```
    max(new_value, old_value);
```

the variables "new_value" and "old_value" are the actual parameters we use. These variables are passed to the function by value. This means that the actual values of the two variables are copied into "x" and "y". We never actually compare the variables "new_value" and "old_value", but copies of them passed in "x" and "y". "x" and "y" are local to the function and have no scope outside its call and are destroyed (de-allocated on the stack) upon return.

For a function to effect a change in the variables outside itself, that variable must either be an external variable, that is one with global scope, or the variable can be passed to the function by reference. In call by reference, the address of the variable in memory is passed and this is used by the function to access the real variable. The address of the variable is a pointer to that variable.

Pointers give C great power, but they can be dangerous in functions if you don't remember that you are operating under call by reference. You may destroy the data you are operating on. Note that functions themselves have global scope for a given compilation unit so that any function can call another. Functions can also call themselves, ie recursion!

Next month we look at pointers and arrays. We will spend some time on pointers as these form the basis of the power of C. We can have pointers to variables, pointers to arrays, pointers to arrays of pointers, pointers to structures, pointers to functions etc...

Variables, Types, Operators and Expressions

These are the building blocks of a language (or Constants aren't, Variables won't).

Variables

No matter what language you use, there are things called Variables where the data you are operating on is held. A variable has several characteristics, those being a Name (or identifier), a Type, a storage class, and a Scope - i.e. Who is it, what's in it, where is it and is it there still?

Variables are a group of things called objects. An object is a region of storage which can have operations performed upon it - also called an "lvalue" in "C". There are two broad classes of object - "simple" and "aggregate", the "aggregate" types being constructed from collections of simple objects.

Variable Identifiers (Names)

A legal Variable identifier in "C" is made up of the alpha-numeric characters (a-z, A-Z, 0-9) plus the underscore "_", and the first character must be a letter. "C" is a case-sensitive language, which means identifiers which differ in case are different identifiers, e.g. "Count" is different to "count" or "COUNT" and refer to different data objects. The number of characters in an identifier name that are significant (that is, the number of characters that are compared before saying that two identifiers are the same) is dependent on the compiler implementation. The original K & R manual states this number to be eight, the proposed ANSI "C" suggests only the first six, most Amiga compilers have at least thirty-one (Lattice 3.10). An added complication is that identifiers used by the linker, ie external variable or function references, may have fewer significant characters than those used within the program code. The practical upshot of this is that when writing code that you hope to be transportable across compiler-linker combinations, identifier length is one of the considerations. The other important restriction on identifier names is that they may not have the same name as the reserved key-words of "C" like "else", "if" etc.

Types

Along with an identifier, an object is associated with a particular type. The "type" of a variable determines the meaning of the data that is held and can be held within it, ie how to interpret the bit-pattern of a region of storage (lvalue), and what operations can be performed on it. The simple objects have one of the "arithmetic types", so called because they can all be interpreted as numbers. There are two sub-classes of "arithmetic types" - "integral types" and "floating types".

The integral object types - char, int, enum

char	The char type can hold any member of the machine's character set. On the Amiga, a "char" is 8 bits (1 byte) long and can store any ASCII character. A "char" can therefore take on any value between 0 and 255 or -128 and +127.
int	The size of "int" is the natural size of the CPU's registers. Under the AmigaDOS environment, this is 32 bits (4 bytes) long. Values between 0 and 4,294,987,295 or -2,147,483,648 and +2,147,483,647.
enum	The enum or enumeration type is a new edition from ANSI "C". It is similar to the type "int", however an object of this type may only take on values from a list of identifiers at its declaration.

The floating object types - float, double

float	Single-precision floating point values may be held by objects of "float". Numbers with fractional parts in the range -1 x 10^{-37} to +1 x 10^{+38}. This number is stored in IEEE format in 32 bits (4 bytes).
double	Double-precision floating point values in the range -1 x 10^{-307} to +1 x 10^{+308}. This is stored in IEEE format in 64 bits (8 bytes).

Pointer objects

Pointer objects are also arithmetic types, but contain numbers that point to other objects. All pointers in the AmigaDOS environment are 32 bits long and are unsigned values between 0 and FFFFFFFF (hexadecimal). Pointer arithmetic differs from that of other arithmetic types and will be covered in detail when we look at pointers.

Type Modifiers

There are also some qualifiers of the basic data types. These qualifiers are called type modifiers and affect the size of the object and the interpretation of its contents.

Size modifiers affect ints:

short int	- or just int is 16 bits (2 bytes) long on the Amiga.
long int	- or just long, is equivalent to int.

Sign modifiers affect char, int:

unsigned	- all bits of object used for magnitude, all numbers positive.
signed	- most significant bit used for sign (+/-) and the rest used to hold magnitude.

Scope and Storage Class

In C, there are four declarable storage classes:

automatic static external register

The storage class determines the location and lifetime of the memory resource associated with a particular identifier. We have said the structure of a C program is made up of blocks (although "C" is not a true block-structured language). Variables will have a certain life within these blocks. The scope of a variable may be either explicit in the variable's definition or implicit by virtue of the location in which the variable was defined.

Automatic variables are created upon entry into the block in which they were defined and are destroyed upon exit from that code block. They are local to each invocation of a block and are undefined outside that block. A variable is automatic by virtue of the fact that it was defined inside a block - implicit scope. A variable may be explicitly defined as being automatic by the use of the storage class modifier "auto". Memory for automatic variables is allocated on the stack. The contents of memory on the stack is unknown and as such the contents of the variable is unknown.

Static variables are local to a block but retain their value on exit from the block and will still have it upon re-entry of that block. Static variables defined within blocks must be explicitly declared to be such by the use of the storage class modifier "static":

```
    static int count;
```

If however, they are defined outside all code blocks, then they are static by implication. These variables are also known as "global" because their identifier is known, ie has scope, throughout the file it is defined in. Memory for static local and global variables as well as external variables, is held in memory obtained from the operating system at the time the program code was loaded. The operating system clears the memory space to zero and in this way, memory that is static is initialized to zero.

External variables exist and retain their values throughout the entire program execution, not simply the source file they were defined in. Thus, they can be used as communication between different blocks and indeed separately compiled pieces of code.

Register variables are (usually, if possible) stored in the fast registers of the CPU. They are usually used for values that must be available quickly - such as loop counters, etc. You can declare any number of variables to be of "register" storage class, but should there be more variables than registers because the excess will be allocated RAM storage locations and there will be no speed advantage.

Thus a storage class, to some extent, defines the scope of the variable throughout the execution of the program.

Declaration of a Variable

Before a variable or identifier can be used, it must be declared to the compiler. This is done at the appropriate place, the beginning of the block in which it will be used. A typical identifier declaration looks something like this:

```
    int count, number, total;
    float result, length;
```

In these two lines, we have given all the information necessary for the compiler to do what it has to when we begin referring to these variables in the program. Here we have defined the variables "count", "number" and "total" to be of type integer. Their scope is confined to the block in which they were declared and their storage class is implicitly static if they were defined outside of any blocks or automatic if inside. Similarly for the floating point variables "result" and "length".

When a variable is defined, it may also be assigned an initial value:

```
    int i = 0;
```

Here, i is set to 0. If the variable is external or static, the initialization is done only once, conceptually before the program starts executing. Automatic variables however, are never initialized. If an automatic variable is not initialized, it contains garbage. External and static variables are initialized to 0 by default, or null in the case of character variables.

Constants

Constants are not really part of the program as such. They appear as #defines. A #define is a compiler directive, and has its effect during compilation rather than during execution of the program. We define a constant before we use it (obviously) at the beginning on the program text like so:

```
    #define CTRLC  '0x3'
```

Here we have defined the identifier "CTRLC" as being a character (single byte) with the hexadecimal value of 3. We know it is a single byte as it is surrounded by single quotes, and this defines a character. The 0x construct tells the compiler that the next digits are to be interpreted as a hexadecimal (base 16) number. The constant is in upper case. It need not be, but this is a tradition in C programming, that all constants are in upper case. Other types of constants are numbers and strings:

```
    #define NAME "John Smith"
    #define MAXLINENUMBER 1000
    #define FORMFEED     '\014'
```

Here we have defined a string constant, surrounded by double quotes, an integer MAXLINENUMBER with its value 1000, and a character FORMFEED defined by the octal construct \xxx where xxx is three octal (base 8) digits. NOTE! a #define is not terminated with a semicolon as it is not a C statement but a compiler directive. We will look at other compiler directives later.

Operators

To be useful, a language must allow us to do things to data. Operators are the means by which we effect this. They allow us to do things to all types of data and storage class.

Arithmetic

The basic arithmetic operators which are found in most languages are + addition, - subtraction, * multiplication, / division. In C there are also the following: % the modulus operator, and a unary -. There is no unary +. The list in order or precedence is: unary -, followed by *,/ and % (all identical), followed by + and binary -. They group left to right and the evaluation of expressions can be altered using parentheses (). A word of warning though, associative and commutative operators * and + may have their order of evaluation altered by the compiler e.g. a+(b+c) may be evaluated as (a+b)+c or a+(b+c). If you have to have a particular order, then the K & R Bible states that you must use temporary variables.

Relational and Logical

The relational operators are: > >= < <=. They have the same precedence. Below them in precedence are the equality operators == (equal to) and != (not equal to). Arithmetic operators have higher precedence than relationals.

The logical operators (or connectives) are && (and), and || (or). These are evaluated left to right and have lower precedence than the relational operators.

Next month, we will continue with operators, looking at some of the nicer features of the language like bit-wise operator and the power (and danger) of type conversion. Hopefully, we will also look at the general overview of a C program and identifying its features and where the things we have spoken about, fit in.

==

Editor's Column
(Written August 6th, 1988)

Hi. Ron Wail and I had an informal meeting with Tony Cuffe of Commodore a week or two ago, when Tony and quite a number of other Commodore staff were in town. One of the more interesting topics was Kickstart/Workbench V1.3. Tony says this should be available at the end of September as a V1.3 Enhancer Pack (or words to that effect) for around $30 or so. The V1.3 Enhancer will include a 1.3 Kickstart disk for 1000 owners, 1.3 Workbench and 1.3 Extras disk, along with a fantastic manual. I've had brief look through the draft of this manual, and it's **great!** All the CLI stuff that has never been explained properly is in there, along with full documentation on all the new programs and stuff you've been reading about in V1.3 articles. For 500 and 2000 owners, new Kickstart ROMS will be available for about $40 or so, just take your machine to a service centre. Note that you **do not** need these new ROMs unless you have a V1.3 compatible auto-booting hard disk drive and want to auto-boot from it.

For developers, a complete set of V1.3 disks and all the notes from the recent developer's conference in the USA will be available in a few weeks. Commodore will notify registered developers by mail about costs and availability.

All Commodore add-on boards for the 2000 (like 68020, huge memory boards, etc) will also be available soon, Commodore apparently already has the boards but is waiting for documentation.

For 2090 hard disk controller owners, Commodore is going to produce an add-on board that will effectively upgrade you to 2090A standard. Either a modified 2090 or a 2090A will allow you to boot your Amiga from the hard disk.

Tony said the Amiga 3000 does not exist. Apparently, anyone who counts has been asked what they would like a 3000 to be if Commodore made one, but that is about as far as it goes. The 2000 is to be Commodore's "standard" Amiga, and the various 2500 models are just 2000s pre-configured by Commodore with boards and drives.

Oops, I seem to have run out of space. I'll see you at the next meeting, August 21st at Monash.

Newsletter Back Issue Order Form

Issue Numbers:								
Number of issues at $2 each								$
Less 10% for 5 or more								- $
Club Use Only								Total $
Mail to: Amiga Users Group, PO Box 48, Boronia, 3155								
Member's Name:								
Address:								

SOFTWARE ORDER FORM

Disk numbers :								
Disks supplied by Amiga User Group @ $8								$
Disks supplied by member @ $2								$
Club Use Only Receipt #: Mailed on: / /								Total $
Mail to: Amiga Users Group, PO Box 48, Boronia, 3155, Victoria.								
Member's Name:								
Address:								

Application for Membership of The Amiga Users Group Inc

Membership is $20 per year. Send your cheque to: **Amiga Users Group Inc, PO Box 48, Boronia, 3155**

Surname: ____________________ Details on this side are optional

First Name: ____________________ Year of birth: ______ Which model Amiga: ______

Address: ____________________ Occupation: ______

____________________ Interests: ______

Phone Number: ______ STD Code: ______ ______

Where did you hear about AUG: ______ ______

____________________ Dealer's Name: ______

____________________ Dealer's Address: ______

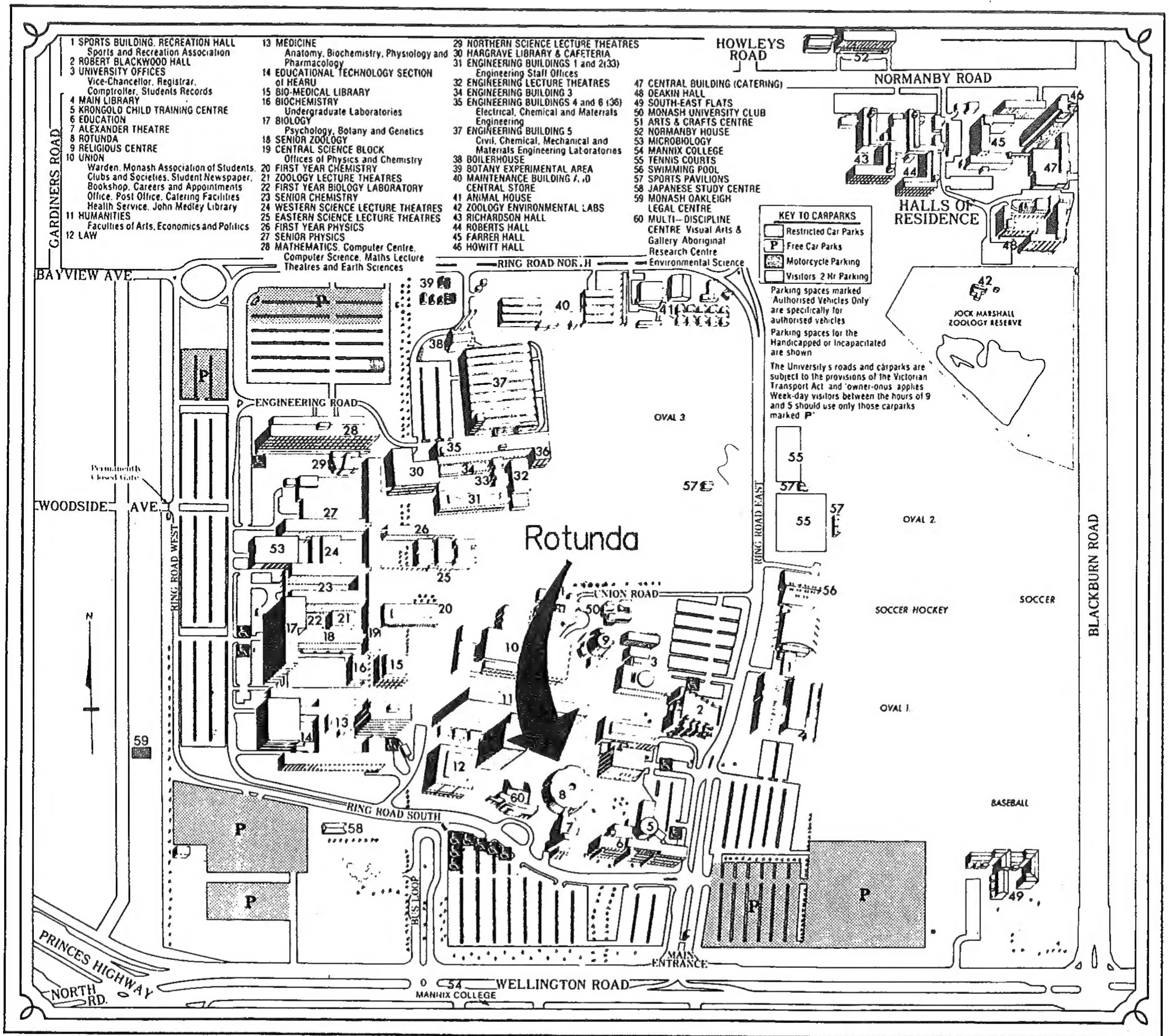
Signed: ______ Date: ______ ______

If admitted as a member, I agree to abide by the rules of the Association for the time being in force.

Club Use Only	Date	Paid	Rcpt #	Memb #	Card Sent

AUG meets on the third Sunday of each month

Monash University is in Wellington Road, Clayton. See Melways Map 70, reference F10. Melways map 84A shows the University Campus in details. I've drawn a huge arrow on the map below to show where the Rotunda is. The best place to park your car is the car park area between Wellington Road and the Rotunda. The entrance to the Rotunda is virtually at the point of the arrow.


1 SPORTS BUILDING, RECREATION HALL Sports and Recreation Association
2 ROBERT BLACKWOOD HALL
3 UNIVERSITY OFFICES Vice-Chancellor, Registrar, Comptroller, Students Records
4 MAIN LIBRARY
5 KRONGOLD CHILD TRAINING CENTRE
6 EDUCATION
7 ALEXANDER THEATRE
8 ROTUNDA
9 RELIGIOUS CENTRE
10 UNION Warden, Monash Association of Students, Clubs and Societies, Student Newspaper, Bookshop, Careers and Appointments Office, Post Office, Catering Facilities Health Service, John Medley Library
11 HUMANITIES Faculties of Arts, Economics and Politics
12 LAW
13 MEDICINE Anatomy, Biochemistry, Physiology and Pharmacology
14 EDUCATIONAL TECHNOLOGY SECTION of HEARU
15 BIO-MEDICAL LIBRARY
16 BIOCHEMISTRY Undergraduate Laboratories
17 BIOLOGY Psychology, Botany and Genetics
18 SENIOR ZOOLOGY
19 CENTRAL SCIENCE BLOCK Offices of Physics and Chemistry
20 FIRST YEAR CHEMISTRY
21 ZOOLOGY LECTURE THEATRES
22 FIRST YEAR BIOLOGY LABORATORY
23 SENIOR CHEMISTRY
24 WESTERN SCIENCE LECTURE THEATRES
25 EASTERN SCIENCE LECTURE THEATRES
26 FIRST YEAR PHYSICS
27 SENIOR PHYSICS
28 MATHEMATICS, Computer Centre, Computer Science, Maths Lecture Theatres and Earth Sciences
29 NORTHERN SCIENCE LECTURE THEATRES
30 HARGRAVE LIBRARY & CAFETERIA
31 ENGINEERING BUILDINGS 1 and 2(33) Engineering Staff Offices
32 ENGINEERING LECTURE THEATRES
34 ENGINEERING BUILDING 3
35 ENGINEERING BUILDINGS 4 and 6 (36) Electrical, Chemical and Materials Engineering
37 ENGINEERING BUILDING 5 Civil, Chemical, Mechanical and Materials Engineering Laboratories
38 BOILERHOUSE
39 BOTANY EXPERIMENTAL AREA
40 MAINTENANCE BUILDING AND CENTRAL STORE
41 ANIMAL HOUSE
42 ZOOLOGY ENVIRONMENTAL LABS
43 RICHARDSON HALL
44 ROBERTS HALL
45 FARRER HALL
46 HOWITT HALL
47 CENTRAL BUILDING (CATERING)
48 DEAKIN HALL
49 SOUTH-EAST FLATS
50 MONASH UNIVERSITY CLUB
51 ARTS & CRAFTS CENTRE
52 NORMANBY HOUSE
53 MICROBIOLOGY
54 MANNIX COLLEGE
55 TENNIS COURTS
56 SWIMMING POOL
57 SPORTS PAVILIONS
58 JAPANESE STUDY CENTRE
59 MONASH OAKLEIGH LEGAL CENTRE
60 MULTI–DISCIPLINE CENTRE Visual Arts & Gallery Aboriginal Research Centre Environmental Science
KEY TO CARPARKS
Restricted Car Parks
P Free Car Parks
Motorcycle Parking
Visitors 2 Hr Parking
Parking spaces marked 'Authorised Vehicles Only' are specifically for authorised vehicles
Parking spaces for the Handicapped or Incapacitated are shown
The University's roads and carparks are subject to the provisions of the Victorian Transport Act and 'owner-onus' applies Week-day visitors between the hours of 9 and 5 should use only those carparks marked P
Rotunda
HOWLEYS ROAD
NORMANBY ROAD
HALLS OF RESIDENCE
GARDINERS ROAD
BAYVIEW AVE
WOODSIDE AVE
Permanently Closed Gate
RING ROAD NORTH
RING ROAD WEST
RING ROAD EAST
RING ROAD SOUTH
ENGINEERING ROAD
UNION ROAD
BUS LOOP
MAIN ENTRANCE
WELLINGTON ROAD
MANNIX COLLEGE
PRINCES HIGHWAY
NORTH RD.
BLACKBURN ROAD
JOCK MARSHALL ZOOLOGY RESERVE
OVAL 1
OVAL 2
OVAL 3
SOCCER HOCKEY
SOCCER
BASEBALL
N

BY PUBLIC TRANSPORT . . . The simplest method is to take a train from Flinders Street or Loop stations on the Dandenong/Pakenham line to either Huntingdale or Clayton. Buses run from these stations to the campus or there is a taxi rank at Clayton. With suitable connections the trip takes about 45 minutes — but it can take longer! An inner neighborhood ticket will take you all the way via Huntingdale station and the bus, but you will need to purchase a comprehensive ticket for the trip via Clayton, which encompasses two neighborhoods. The campus is also served by buses from Box Hill, Blackburn, Belgrave, Chadstone, Jells Park-Glen Waverley, Dandenong-Mulgrave, Oakleigh and Elwood.

FROM THE CITY BY CAR . . . An easy route is along St Kilda Road or Kingsway/Queens Road and then on to Dandenong Road. The campus's tall Menzies Building comes into view a kilometre or so before the left turn into Wellington Road on which the main entrance is located. Allow 40-50 minutes for the trip. Drivers should note that restrictions apply in some car parks weekdays 9 a.m. to 5 p.m. and fines **do** apply. There is ample unrestricted parking and, closer to buildings, designated two hour visitor car parks — check the map or ask at the Gatehouse.